(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2014-211725
(P2014-211725A)
(43)公開日　平成26年11月13日(2014.11.13)

(51)Int.Cl.		F I		テーマコード（参考）
G06N 5/04	(2006.01)	G06N	5/04	570C

審査請求 未請求　請求項の数 29　ＯＬ　（全 27 頁）

(21)出願番号　特願2013-87008(P2013-87008)
(22)出願日　平成25年4月17日(2013.4.17)

特許法第３０条第２項適用申請有り　第４８回石川県発明くふう展（平成２４年１０月１７日～２１日）

(71)出願人　301062798
ナレルシステム有限会社
東京都板橋区高島平９－３－８－２０２号
(74)代理人　300004728
中村　圭介
(72)発明者　中村　圭介
東京都板橋区高島平９丁目３番８号ナレルシステム有限会社内

(54)【発明の名称】知識や情報を処理する方法、装置及びコンピュータプログラム

(57)【要約】　（修正有）
【課題】自然言語に変数を埋め込む形式で表現した知識をコンピュータでＰＲＯＬＯＧ同様に自動処理することを実現可能する。
【解決手段】人間が、ＰＲＯＬＯＧにおける「事実」、「ルール」又は「ゴール」の入力にあたり、ＰＲＯＬＯＧにおける「リテラル」にあたる内容の定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別してコンピュータに入力し、コンピュータが、文の主部、述部等の境界をまたがりうるものとして変数を取扱いながら、その入力に含まれる文の自動単一化処理、自動導出処理又はその両方を行う。
【選択図】図１

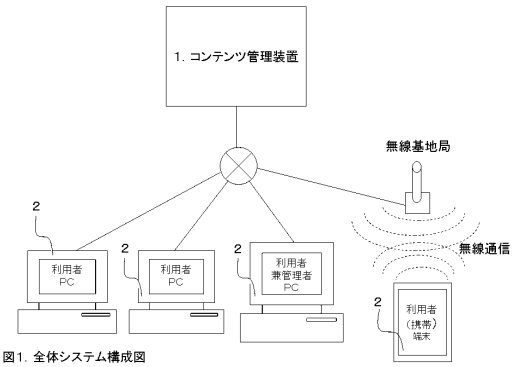

図1. 全体システム構成図

【特許請求の範囲】
【請求項１】
コンピュータが、論理型プログラミング言語ＰＲＯＬＯＧの機能の一部又は全部を自動的に行う方法であって、
人間が、ＰＲＯＬＯＧにおける「事実」、「ルール」又は「ゴール」の入力にあたり、ＰＲＯＬＯＧにおける「リテラル」にあたる内容（以降、「文」と呼ぶ）の定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別してコンピュータに入力すること、
コンピュータが、文の主部、述部、主語、述語、目的語、補語、連用修飾語、連体修飾語又は文の境界をまたがりうるものとして変数を取扱いながら、上記「事実」、「ルール」又は「ゴール」の入力に含まれる文の自動単一化処理、自動導出処理又はその両方を行うこと、
を特徴とする処理方法。
【請求項２】
前記自動単一化処理において、
変数を含みうる文であるパターン（定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別したもの）と、変数を含まない文である定数文字列との単一化を、
パターン（例：＄Ｘと＄Ｙは＄Ｚが好き）に最初（又は最後）に出現する変数（例：＄Ｘ）の最長一致解（例：太郎と次郎）を求めて、該最長一致解の文字列を後ろ（又は前）から空文字になるまで削る過程（例：太郎と次郎＞太郎と次＞太郎と＞太郎＞太＞空）を該パターンに代入してできる各新たなパターン（例：「太郎と次郎と＄Ｙは＄Ｚが好き」＞「太郎と次と＄Ｙは＄Ｚが好き」＞「太郎とと＄Ｙは＄Ｚが好き」＞「太郎と＄Ｙは＄Ｚが好き」＞「太と＄Ｙは＄Ｚが好き」＞「と＄Ｙは＄Ｚが好き」）について、該新たなパターンができる前提となった削る過程の解の各状態（＄Ｘ＝太郎と次郎＞太郎と次＞太郎と＞太郎＞太＞空）をそれぞれ記憶しながら、
再帰的に同様に変数がなくなるまで繰り返し、
変数がなくなったときに単一化した各途中の解の状態（例：真（＄Ｘ＝太郎と次郎，＄Ｙ＝三郎），真（＄Ｘ＝太郎，＄Ｙ＝次郎と三郎））を結合して各単一化の答え（例：答え１＝（＄Ｘ＝太郎と次郎，＄Ｙ＝三郎），答え２＝（＄Ｘ＝太郎，＄Ｙ＝次郎と三郎）を得ることを特徴とする
請求項１に記載の処理方法。
【請求項３】
空文字になるまで徐々に削るかわりに、別途求めた最短一致解までしか徐々に削らないことを特徴とする
請求項２に記載の処理方法。
【請求項４】
空文字になるまで徐々に削る際、
パターン中で該変数が最初（又は最後）に出現した直後（又は直前）の固定文字である１文字Ｃ（直後（又は直前）が別の変数でその解が空文字でないときは該別の変数に解候補として代入されている解候補の最初（又は最後）の１文字Ｃ）が該最長一致解に含まれる場合に、
該最長一致解の文字列の最後（又は最初）からそのＣに最初にあたるまでの部分文字列をそのＣを含め削ることを特徴とする
請求項２又は３に記載の処理方法。
【請求項５】
前記自動単一化処理又は自動導出処理において、
前記「ゴール」（「サブゴール」を含む。以下同じ）となる束縛情報を求める際、
該「ゴール」を構成する各文について独立して束縛情報を求め
各束縛情報の積集合が空でない場合に戻り値真とその積集合を返す
請求項１から４のいずれか一項に記載の処理方法。

【請求項６】
前記自動単一化処理又は自動導出処理において、
前記「ゴール」となる束縛情報を求める際、
既存の束縛情報を最初に空とし、
「ゴール」が含む文集合を、既存の束縛情報の適用下で該一文が真となりうるかと該なりうるための追加の束縛情報について自動計算して該追加の束縛情報を該既存の束縛情報に加えながら該一文を除いて再帰的に小さくしていき、
真となりうるまま該文集合が空集合になったときの既存の束縛情報を「ゴール」となる束縛情報とする
請求項１から４のいずれか一項に記載の処理方法。
【請求項７】
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンと変数を含まない事実との単一化を行う場合、
パターンと事実が文字列として一致する場合に真を返す、又は
パターンと事実が文字列として一致しない場合に偽を返す
請求項１から６のいずれか一項に記載の処理方法。
【請求項８】
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンとホーン節の変数を含まないヘッド部との単一化を行う場合、
パターンとヘッド部が文字列として一致する場合に、該ホーン節のボディ部をサブゴールとして請求項５又は６に記載の処理方法を適用した結果を返し、
一致しない場合に偽を返す
請求項１から７のいずれか一項に記載の処理方法。
【請求項９】
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「定数文字列」として当該パターンを用いかつ同処理方法における「パターン」として当該ヘッド部を用いて処理して求めた各束縛情報について、
当該ホーン節のボディ部に適用したものを請求項５又は６に記載の処理方法の「サブゴール」として処理した結果が真となるものが存在すれば真を返す、又は
存在しなければ偽を返す
請求項１から８のいずれか一項に記載の処理方法。
【請求項１０】
前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含まない事実との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「パターン」として当該パターンを用いかつ同処理方法における「定数文字列」として当該事実を用いて処理した結果を返す、
請求項１から９のいずれか一項に記載の処理方法。
【請求項１１】
前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含まないヘッド部をもつホーン節との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「パターン」として当該パターンを用いかつ同処理方法における「定数文字列」として当該ヘッド部を用いて処理した結果を真とする束縛情報（Ｃ）が存在する場合で、かつ、当該ホーン節のボディ部を請求項５又は６に記載の処理方法の「サブゴール」として処理した結果が真の場合に当該束縛情報（Ｃ）とともに真を返す、
請求項１から１０のいずれか一項に記載の処理方法。
【請求項１２】

前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
パターンとヘッド部の頭、尾又は両方に存在する文字数分の固定文字が一致していない場合に偽を返す、
請求項１から１１のいずれか一項に記載の処理方法。
【請求項１３】
前記自動単一化処理又は自動導出処理において、
変数を含む第一のパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
パターンとヘッド部の頭及び尾に存在する文字数分の固定文字が一致している場合、
請求項５又は６に記載の処理方法における「サブゴール」として当該ホーン節のボディ部を適用した結果が真となる各束縛情報により当該ホーン節のヘッド部を束縛して得られる各第二のパターンについて、
第二のパターンが変数を含まない場合に、
請求項２から４のいずれか一項に記載の処理方法における「パターン」として第一のパターンを用いかつ同処理方法における「定数文字列」として第二のパターンを用いて処理した結果を返す、
請求項１から１２のいずれか一項に記載の処理方法。
【請求項１４】
変数を含む事実が入力されることをコンピュータが制限することを特徴とする
請求項１から１３のいずれか一項に記載の処理方法。
【請求項１５】
前記ルールがいわゆるホーン節と同様の形式であり、ヘッド部がボディ部に出現しない変数を含むことをコンピュータが制限することを特徴とする
請求項１から１４のいずれか一項に記載の処理方法。
【請求項１６】
前記自動単一化処理又は自動導出処理において、
束縛されない変数が互いに残っているパターンとヘッド部又は事実との自動単一化処理を、
固定文字列として極小となる解の集合を求めて返すことにより行うことを特徴する、
請求項１から１３のいずれか一項に記載の処理方法。
【請求項１７】
前記自動単一化処理又は自動導出処理において、
束縛されない変数が互いに残っているパターンとヘッド部又は事実との自動単一化処理を、
変数を含む文字列として極小となる解の集合を求めて返すことにより行うことを特徴する、
請求項１から１３のいずれか一項に記載の処理方法。
【請求項１８】
束縛されない変数が互いに残っているパターンとヘッド部又は事実との各文字列の頭及び尾に共通して存在する文字数分の固定文字が一致していない場合、
偽を返すことを特徴とする、
請求項１６又は１７に記載の処理方法。
【請求項１９】
前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、
束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、

各変数に相手の文字列（第一の文字列中の変数には第二の文字列、第二の文字列中の変数には第一の文字列）のあらゆる部分文字列（＜空＞及び文字列全体も含む）又は自分自身を代入してみて両文字列が一致する場合を探す全探索により極小となる解の集合を求めることを特徴する、
請求項１６又は１７に記載の処理方法。
【請求項２０】
束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列そのものではなく、
束縛されない変数が互いに残っているパターンとヘッド部又は事実との各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除いて残る二つの文字列に相当する両文字列について、
請求項１９に記載の処理をする処理方法。
【請求項２１】
各変数に相手の文字列のあらゆる部分文字列（＜空＞及び文字列全体も含む）又は自分自身を代入してみて両文字列が一致する場合を探す全探索により極小となる解の集合を求める際、
変数に代入する部分文字列に変数が含まれる場合に、当該部分文字列に含まれる変数を両文字列にそれまで存在しなかった新しい変数で置き換えながら、該新しい変数に対しての同様の代入も以後再帰的に行いながら両文字列が一致する場合を探す全探索を行うことを特徴とする、
請求項１９又は２０に記載の処理方法。

【請求項２２】
変数に代入した結果についても、各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除いて残る二つの文字列を、
「相手の文字列のあらゆる部分文字列（＜空＞及び文字列全体も含む）」の対象となるの「相手の文字列」として、探索が進むごとに、文字列の頭又は尾に存在する固定文字数が少なくすることを特徴とする、
請求項１９～２１のいずれか一項に記載の処理方法。
【請求項２３】
前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、
束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、
各変数に、固定文字列（空文字を含む）又は変数を含む文字列を再帰的に代入してみて解の集合を求める処理方法であって、
代入してみる変数を相手文字列の頭又は尾が固定文字である自文字列の対応する頭又は尾の変数に限定し、
代入値を、該頭である場合には該頭の固定文字もしくは固定文字列に続く新しい変数又は＜空＞とし、該尾である場合には新しい変数に続く該尾の固定文字もしくは固定文字列又は＜空＞とし、
変数に代入した結果について、各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除くことを繰り返す、
ことを特徴とする請求項１６又は１７に記載の処理方法。
【請求項２４】
前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、
束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛

され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、
各変数に、固定文字列（空文字を含む）を再帰的に代入してみて解の集合を求める処理方法であって、
各文字列の頭及び尾に存在する文字数分の共通する固定文字を除いてできる両文字列中の固定文字種（変数の識別子やデリミタを除いた文字種）だけを用いてできる束縛情報を、束縛する値の文字列長の合計が小さいものから順に列挙していくことにより求める、
請求項１６又は１７に記載の処理方法。
【請求項２５】
求めた「変数を含む文字列として極小となる解の集合」をもとに、
当該ホーン節の変数を当該変数を含む文字列で一時的に書き換えてできるホーン節により探索を継続する請求項１９～２４に記載の処理方法。
【請求項２６】
求めた「変数を含む文字列として極小となる解の集合」をもとに、
「変数を含む文字列」の変数部分に、当該「変数を含む事実」の解組の集合としてあらかじめ定義された固定文字列組の集合の各要素を代入してできる固定文字列組の集合を返す、
請求項１９～２５のいずれか一項に記載の処理方法。
【請求項２７】
前記自動単一化処理又は自動導出処理において、
変数を含む第一のパターンと変数を含む第二のパターンとの自動単一化を行う場合、
両パターンの頭及び尾に存在する文字数分の固定文字が一致しており、かつ、同じ変数が各パターン内で一度しか出現しない場合、
無限個の解の存在を意味する情報を返す、
請求項１から２６のいずれか一項に記載の処理方法。
【請求項２８】
請求項１から２７のいずれか一項の方法を実行するための装置。
【請求項２９】
請求項１から２７のいずれか一項の方法をコンピュータに実行させるためのコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、知識や情報を処理する方法、特に自然言語で表現された知識や情報をその自然言語の「まま」処理する方法、装置及びコンピュータプログラムに関するものである。
【背景技術】
【０００２】
知識や情報を処理する方法、特に自然言語で表現された知識や情報をその自然言語の「まま」処理する方法としては、自然言語で表現された知識や情報をもとに論理型プログラミング言語ＰＲＯＬＯＧと同様の述語論理レベルの演繹（例えば、三段論法等によりＹＥＳ／ＮＯ形式の問い（いわゆるクローズドクエスチョン形式）に答えること）や解探索（例えば、指定した条件（いわゆるオープンクエスチョン形式）に合致する解の集合を求めること）や手続的解釈、等を、ＰＲＯＬＯＧと同様の自動単一化処理（ユニフィケーション）や自動導出処理（バックトラックを伴う後ろ向き推論。なお、導出過程を表示した場合は自動証明処理となる）等によって行おうとする下記特許文献１がある。
この特許文献に記載された技術内容の実現により、利用者が、自分の言葉で普通に（オントロジーや表現記法の標準化に配慮せず、格の出現順序も自由に）作った文章や、世界中の人々がネットワーク上に公開した自然言語による記事を普通に引用したり参照したり（例えば、コピーペーストやＵＲＬ指定する等）するだけで、その元の自然言語のまま、

利用者による演繹や解探索等（すなわち、自動単一化処理、自動導出処理、等）の知識源とすることができるようになる。こうして、この特許文献１により、全く形式化されていない自然言語レベルの知識や情報の、一般人（例えば、数学の授業で「変数」を教えられた中学生程度以上ではあるが、情報処理や知識処理の専門家ではない人々）による利用・応用について、格段の進歩が期待されていた。

【０００３】

しかし、上記特許文献１の技術の実現には困難な点がいくつかあった。すなわち、「動詞（主語，目的語，いつ，どこで）」等と格構造が「，」（コンマ）で区切られている（したがって、「引数」の順番によりそれぞれの役割（格）が決まっている）従来のＰＲＯＬＯＧとは異なり、「Ｘ」（全角の大文字（文字種）による変数識別子の例）、「＄Ｘ」（半角や全角のエスケープ文字（＄）と変数識別子（Ｘ）の例）、「＄｛行為主体｝」（エスケープ文字（＄）とデリミタ（｛・・・｝）と変数識別子「行為主体」の例）等として表現された変数を自然言語の文中に埋め込んで「＄Ｘと＄Ｙは＄Ｚが好き」等と表現した場合、「太郎と次郎と三郎はカレーが好き」を根拠とした変数組（＄Ｘ，＄Ｙ，＄Ｚ）の解候補は、コンピュータが一通り「だけ」挙げればよいのではなく、「（＄Ｘ，＄Ｙ，＄Ｚ）＝（太郎と次郎，三郎，カレー）及び（太郎，次郎と三郎，カレー）」等と複数の解候補を列挙してそれぞれの解としての可能性を検査・検証できなければ、知識源からコンピュータが網羅的に演繹したり解探索したりしたこと（すなわち、ＰＲＯＬＯＧにおけるユニフィケーション（自動単一化処理）やバックトラックを伴う後ろ向き推論（自動導出処理）を完全に行ったこと）にはならない。

【０００４】

つまり、「太郎と次郎と三郎はカレーが好き」等といった一つの事実（ＰＲＯＬＯＧにおけるファクト「ＦＡＣＴ」）に対して、前述した特許文献１で問題にした着眼点（表記ゆれの吸収の為等）とは異なる（しかも、ＰＲＯＬＯＧの通常のバックトラックによって複数の解候補を網羅するため（それは、あくまで格構造に縛られた範囲を網羅するため）に必要ないずれの着眼点とも当然に異なる）着眼点で複数の単一化の可能性がでてくるそのすべての単一化をコンピュータが試してみることが必要になる。

【０００５】

これは、自然言語で表現された事実（変数を含む事実（例えば、「＄Ｘは＄Ｘである。」）又は含まない事実（例えば、「太郎と次郎と三郎はカレーが好き。」））との自動単一化処理だけではなく、ＰＲＯＬＯＧのホーン節（ルール）のヘッド部に相当する（多くの場合に束縛すべき変数を含むような）自然言語表現との単一化（すなわち、変数を含みうるもの同士の単一化）や、その単一化による束縛値（例：Ｘ←太郎と次郎、等）のボディ部の変数（スコープが同じである変数）への波及について、網羅的な自動導出処理のためのバックトラックとの関係上さらに複雑な問題となるため、自然言語に変数を埋め込む形式で表現した知識（事実（＝ファクト、「ＦＡＣＴ」）やルールやゴール）をＰＲＯＬＯＧ同様に自動処理することを困難にしていた（「困難性Ａ」と呼ぶ）。

また、特許文献１におけるような、表記ゆれに対するロバスト性に考慮した応用的な実装（文の表層的／意味的な類似性を根拠とした知識の導出過程への類推適用）や、現状の多くの自然言語システムで導入されているハイブリッドな知識表現の在り方（書換／推論ルール、辞書（意味論、語用論、等）、コーパス分析（ｎ－ｇｒａｍ統計、共起頻度、共起確率表現、等）の組合せ）は、推論結果の基となった知識集合の見やすい表示や導出の流れの直観的なトレースを複雑／困難にし、述語論理学の学習用途や個人的な哲学・ポリシー等の研究用途のためには、役立ちにくいものになっていた。

【先行技術文献】

【特許文献】

【０００６】

【特許文献１】特開２００６－０２４０４５号公報

【発明の概要】

【発明が解決しようとする課題】

【０００７】
したがって、本発明が解決しようとする課題は、課題１）上記した実現上の「困難性Ａ」を克服して、自然言語に変数を埋め込む形式で表現した知識をコンピュータでＰＲＯＬＯＧ同様に自動処理することを実施可能にし、これにより自然言語を知識源として述語論理レベルの網羅的な演繹や解探索（自動単一化処理、自動導出処理、そのための後ろ向き推論、バックトラック、等）を実現することである。ここで、自動導出処理には、ボディ部単一化後のヘッド部変数の書換が必要となり、さらにボディ部の手続的解釈まで行う場合はヘッド部及びボディ部における「より左」のリテラルの単一化に応じて「より右」のリテラル内（さらに、「手続的解釈」でコマンド操作を行う場合は「コマンド」内も）の変数の書き換えが必要となる場合がある。
【０００８】
また、別の課題は、課題２）変数が格構造に縛られないという仕様の利点をより広範に生かし、課題３）格構造に縛られない（ある意味、自由であるが故に過酷な）条件下での処理を高速化し、課題４）学習用途や研究用途等のために根拠（証明過程、推論パス、用いた知識集合、等）を見やすく表示することである。
【課題を解決するための手段】
【０００９】
本発明は、かかる課題に鑑みてなされたものであり、請求項１は、
コンピュータが、論理型プログラミング言語ＰＲＯＬＯＧの機能の一部又は全部を自動的に行う方法であって、
人間が、ＰＲＯＬＯＧにおける「事実」、「ルール」又は「ゴール」の入力にあたり、ＰＲＯＬＯＧにおける「リテラル」にあたる内容（以降、「文」と呼ぶ）の定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別してコンピュータに入力すること、
コンピュータが、文の主部、述部、主語、述語、目的語、補語、連用修飾語、連体修飾語又は文の境界をまたがりうるものとして変数を取扱いながら、上記「事実」、「ルール」又は「ゴール」の入力に含まれる文の自動単一化処理、自動導出処理又はその両方を行うこと、
を特徴とする処理方法を提供する。
ここで、「人間が、・・・定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別してコンピュータに入力する」とは、人間が本発明によるアプリケーションソフト（処理系）の入力欄内やコマンドプロンプト後等に（フロントエンドプロセッサ等を介して又は介さずに）直接入力すること、インターネットブラウザや文章編集ソフト等からコピーペーストやカットペーストすること、別途入力して保存してあったテキストファイル等を読み込んだり、引用したり、参照したりすること等を含む。
また、「コンピュータが、文の主部、述部、主語、述語、目的語、補語、連用修飾語、連体修飾語又は文の境界をまたがりうるものとして変数を取扱い」とは、日本語等の助詞等や英語等の前置詞等を介した明示的な区切りのある格構造（構文解析結果や意味解析結果の階層構造に該当するものを含みうる）の境界（明示的な区切り）をまたがりうるだけでなく、英語等の例えばＳＶ形式・ＳＶＯ形式・ＳＶＯＯ形式・ＳＶＯＣ形式のＳとＶの境界、ＶとＯの境界、ＶとＣの境界等をまたがり得、重文や複文の文の境界等をもまたがり得るものとして取り扱うモードを含む（逆に、取り扱わないモードがあってもよい）。これらの場合、変数に入り得る解候補として、文全体や、文全体を構成する文字列から文としての文法構造による区切りの単位を無視して任意の部分文字列を抽出したもの、さらには階層的な参照構造をもつ複文・重文やもっと長い論文等の全体や、さらにそれらの任意の部分文字列が対象となる（なお、「文の境界」をもまたがりうるものとして取り扱う場合であって、知識源の集合を扱う（例えば、ファイルにまとめた知識源の集合を一括して読み込む、書き込む、参照する、等する）場合には、文の境界「以外」に知識源どうしの区切りが必要となるため、このような区切りとして、例えば、改行コードやタブコード等を用いることができる）。

これにより、自然言語に変数を埋め込む形式で表現した知識や情報も、普通の文章形式の知識や情報も、それらの間での現実的な相互活用の可能性が十分に開けた一元的な知識源として、両者を特に区別せずにコンピュータで自動処理することが可能になる（上記「困難性Ａ」の克服）。

「現実的」な相互作用の可能性が「十分」に開けるためには、コンピュータによる自動処理が検索処理系として備えるべき１）検索式（パターンを指定する表現）の簡便性、２）十分な適合率（解の正確性）、３）十分な再現率（解候補を抽出する上での網羅性）を備えていなければならないが、本発明では、文中の自由な（すなわち、文法や意味の区切りに制限されない）場所に変数（複数種類でもよく複数回でもよい）の埋め込みを許容し（簡便性を確保し）つつも、ＰＲＯＬＯＧの知識処理手順（と内在する知識解釈方針）に基本的に沿うことにより、他の処理系におけるようなハイブリッドな知識処理手順（と、そのハイブリッドな処理手順に対応する複雑な知識解釈方針）によって起こりうる適合率（解の正確性）低下のリスクを回避しつつ、「文の主部、述部、主語、述語、目的語、補語、連用修飾語、連体修飾語又は文の境界をまたがりうるものとして変数を取扱」うことにより、自然言語の区切り表現（読点、句読点、カッコ、コンマ、ピリオド、等）や修飾的なフレーズ（形容詞句、副詞句、必須でない格の表現、等）による指定パターンの泣き別れ出現が原因となる不当な不一致（正規表現的なアンマッチ）を防ぎ、格構造の明示されない表層的な文字列表現間（検索式と知識源の間）での一定の再現性（解候補を抽出する上での網羅性）を確保可能としている。また、知識源となった自然言語の正確な文脈（前後の文字列）を損ないにくい一体的・連続的な引用・抽出（変数への解候補としての）も可能となり、結局、その透明性・単純性により推論の実質的な健全性（根拠や推論経緯の正当性）の確認・検証が手軽かつ容易となり、根拠がいつも明確であるという現実的な安心感を利用者にもたらし、全体として、変数が格構造に縛られない（さらに、格構造のルール（出現順序等）について知らなくてもよく、引数の順番が間違っているかもしれないという不安もない）という仕様の利点を十分に生かしきることを可能にする。

なお、本発明は日本語（一文字の情報量が多く、単語の間にスペースがない）を第一の対象にしているため、文字を処理単位、区切りの単位としている内容が多いが、英語等（一文字の情報量が少なく、一文の文字数は多いが、単語の間にスペースのあるもの）は、スペースで区切った単語を単位として同様の処理をすることも可能である。

【００１０】

また、請求項２は、

前記自動単一化処理において、

変数を含みうる文であるパターン（定数部分と変数部分とを文字種、デリミタ又はエスケープ文字により区別したもの）と、変数を含まない文である定数文字列との単一化を、

パターン（例：＄Ｘと＄Ｙは＄Ｚが好き）に最初（又は最後）に出現する変数（例：＄Ｘ）の最長一致解（例：太郎と次郎）を求めて、該最長一致解の文字列を後ろ（又は前）から空文字になるまで削る過程（例：太郎と次郎＞太郎と次＞太郎と＞太郎＞太＞空）を該パターンに代入してできる各新たなパターン（例：「太郎と次郎と＄Ｙは＄Ｚが好き」＞「太郎と次と＄Ｙは＄Ｚが好き」＞「太郎とと＄Ｙは＄Ｚが好き」＞「太郎と＄Ｙは＄Ｚが好き」＞「太と＄Ｙは＄Ｚが好き」＞「と＄Ｙは＄Ｚが好き」）について、該新たなパターンができる前提となった削る過程の解の各状態（＄Ｘ＝太郎と次郎＞太郎と次＞太郎と＞太郎＞太＞空）をそれぞれ記憶しながら、

再帰的に同様に変数がなくなるまで繰り返し、

変数がなくなったときに単一化した各途中の解の状態（例：真（＄Ｘ＝太郎と次郎，＄Ｙ＝三郎），真（＄Ｘ＝太郎，＄Ｙ＝次郎と三郎））を結合して各単一化の答え（例：答え１＝（＄Ｘ＝太郎と次郎，＄Ｙ＝三郎），答え２＝（＄Ｘ＝太郎，＄Ｙ＝次郎と三郎）を得ることを特徴とする

請求項１に記載の処理方法を提供する。

ここで、「（又は最後）」は「（又は前）」に対応しており、最後に出現する変数から代入するときは、削り始めが前からとなる必要があることを意味する。

これにより、最長一致解から最短一致解に至るすべての可能性について確実に網羅した各解候補の状態を検証し、網羅的な答え（さらに絞りこまれた解候補の集合）を提供することが可能になる。

【００１１】

また、請求項３は、

空文字になるまで徐々に削るかわりに、別途求めた最短一致解までしか徐々に削らないことを特徴とする

請求項２に記載の処理方法を提供する。

これにより、最短一致解未満の長さから「空」文字までの解探索の計算を節約することができ、自動処理の高速化が可能となる場合がある。

【００１２】

また、請求項４は、

空文字になるまで徐々に削る際、

パターン中で該変数が最初（又は最後）に出現した直後（又は直前）の固定文字である１文字Ｃ（直後（又は直前）が別の変数でその解が空文字でないときは該別の変数に解候補として代入されている解候補の最初（又は最後）の１文字Ｃ）が該最長一致解に含まれる場合に、

該最長一致解の文字列の最後（又は最初）からそのＣに最初にあたるまでの部分文字列をそのＣを含め削ることを特徴とする

請求項２又は３に記載の処理方法を提供する。

ここで、「（又は最後）」は「（又は直前）」に対応しており、最後に出現する変数から代入するときは、削り始めが前からとなる必要があることを意味する。

これにより、変数の直後（又は直前）の固定文字にマッチしない解候補の列挙とその解候補を前提とした探索を節約することができ、自動処理の高速化が可能となる場合がある。

【００１３】

また、請求項５は、

前記自動単一化処理又は自動導出処理において、

前記「ゴール」（「サブゴール」を含む。以下同じ）となる束縛情報を求める際、

該「ゴール」を構成する各文について独立して束縛情報を求め

各束縛情報の積集合が空でない場合に戻り値真とその積集合を返す

請求項１から４のいずれか一項に記載の処理方法を提供する。

ここで、束縛情報とは、各変数に同時にそのような解候補を代入してみたか／すべきかということを示す情報である。各変数への一つの束縛の仕方を示す束縛情報は、「，」をＡＮＤ結合と解釈することにより、「変数名１＝値１，変数名２＝値２，・・・，変数名Ｎ＝値Ｎ」といったリスト形式で表現することができる一方、複数の束縛の仕方は、そのようなリストの集合として、あるいは、「，」ではなくＡＮＤ／ＯＲ結合を明示して複数束縛間の共通部分をくくりだした形や、複数束縛の全体を標準化した連言標準形や選言標準形で表現することができる。

これにより、同時に満たすべきパターン（文）が複数あり、パターン（文）間で同名の変数には同じ解が入るべきとする問題形式の場合（例えば、ユーザからのそのようなＡＮＤ結合問合せや、ルールのボディ部に複数の文がある場合）に、ルールのより右側のパターンについて、より左側のパターンの各単一化結果ごとに多大な全文探索処理を無駄に繰り返す可能性がすくなくなり、ボディ部をバックトラックしながら深さ優先探索する非効率（変数を自然言語に埋め込む形式においてより顕著な非効率）を回避し、同時に満たすべき各パターンごとの処理を単純化することができる。また、推論過程の表示や検証（完全性や健全性）をわかりやすく、容易にする。また、ルールの意味として必要な直観にとって余計であり定義されるべきでなかった順序性（ボディ部の文間の）が推論過程に副作用を与えることを防ぐことができる。

【００１４】

また、請求項６は、
前記自動単一化処理又は自動導出処理において、
前記「ゴール」となる束縛情報を求める際、
既存の束縛情報を最初に空とし、
「ゴール」が含む文集合を、既存の束縛情報の適用下で該一文が真となりうるかと該なりうるための追加の束縛情報について自動計算して該追加の束縛情報を該既存の束縛情報に加えながら該一文を除いて再帰的に小さくしていき、
真となりうるまま該文集合が空集合になったときの既存の束縛情報を「ゴール」となる束縛情報とする
請求項１から４のいずれか一項に記載の処理方法を提供する。
ここで、「再帰的に小さくしていき」とは、再帰が深くなるほど文集合が小さくかつ束縛情報が複雑になる（より多くの変数が束縛され、束縛する値の範囲がより絞られる）よう、例えば、Ｃ言語で再帰呼び出しをする処理や、独自に宣言し定義した深さ優先探索用のスタック構造（構造体の配列とスタックポインタ等）によって同等の再帰的処理を実現することをいう。
これにより、変数を自然言語に埋め込む形式で表現された、同時に満たすべき複数パターンのＡＮＤ結合からなるオープンクエスチョンを自動処理する問題を、より単純かつ再帰的な同一構造を持つ部分問題へとシンプルに変換し、推論過程の表示や検証（完全性や健全性の）を容易にする。

【００１５】

また、請求項７は、
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンと変数を含まない事実との単一化を行う場合、
パターンと事実が文字列として一致する場合に真を返す、又は
パターンと事実が文字列として一致しない場合に偽を返す
請求項１から６のいずれか一項に記載の処理方法を提供する。
これにより、シンプルかつ迅速に探索を行うことができる。
なお、「変数を含まない事実」が句点等で区切られた複数の文を含む場合。「文字列としての一致」は、パターンの文字列が事実の文字列の部分に一致することであってもよいとするモードを設けることもできる。
これにより、同型の複数の情報を一つの事実にまとめて表現して探索の対象とすることができる。また、複数要素間のなんらかの順序性（例えば、大きさ、貴重さ、古さ・・・等）も表現しやすくなる。

【００１６】

また、請求項８は、
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンとホーン節の変数を含まないヘッド部との単一化を行う場合、
パターンとヘッド部が文字列として一致する場合に、該ホーン節のボディ部をサブゴールとして請求項５又は６に記載の処理方法を適用した結果を返し、
一致しない場合に偽を返す
請求項１から７のいずれか一項に記載の処理方法を提供する。
これにより、変数を自然言語に埋め込む形式で表現するルールのヘッドが変数を含まない場合の健全な機械的解釈と、シームレスでわかりやすくかつシンプルで統一的な処理メカニズムと、対応する推論過程の説明を与えることができる。

【００１７】

また、請求項９は、
前記自動単一化処理又は自動導出処理において、
変数を含まないパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「定数文字列」として当該パターンを用いかつ同処理方法における「パターン」として当該ヘッド部を用いて処理して

求めた各束縛情報について、
当該ホーン節のボディ部に適用したものを請求項５又は６に記載の処理方法の「サブゴール」として処理した結果が真となるものが存在すれば真を返す、又は
存在しなければ偽を返す
請求項１から８のいずれか一項に記載の処理方法を提供する。
これにより、変数を自然言語に埋め込む形式で表現するルールのヘッドが変数を含み、このルールを起動したパターンが変数を含まない場合に、ボディ部の変数の束縛範囲を事前に制限して、真偽判断の処理を高速化することができる。
また、健全な機械的解釈と、シームレスでわかりやすくかつシンプルで統一的な処理メカニズムと、対応する推論過程の説明を与えることができる。

【００１８】
また、請求項１０は、
前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含まない事実との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「パターン」として当該パターンを用いかつ同処理方法における「定数文字列」として当該事実を用いて処理した結果を返す、
請求項１から９のいずれか一項に記載の処理方法を提供する。
これにより、パターンと（静的な）事実との間のすべての解候補（格変数への）が洗い出されて「結果」として返されるため、上記「困難性Ａ」克服のための基礎的な機構をシンプルに与えることができ、推論過程の可視化や検証だけでなくさらにはルールの修正等といったユーザーの手を介した学習をも容易にする。

【００１９】
また、請求項１１は、
前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含まないヘッド部をもつホーン節との単一化を行う場合、
請求項２から４のいずれか一項に記載の処理方法における「パターン」として当該パターンを用いかつ同処理方法における「定数文字列」として当該ヘッド部を用いて処理した結果を真とする束縛情報（Ｃ）が存在する場合で、かつ、当該ホーン節のボディ部を請求項５又は６に記載の処理方法の「サブゴール」として処理した結果が真の場合に当該束縛情報（Ｃ）とともに真を返す、
請求項１から１０のいずれか一項に記載の処理方法を提供する。
これにより、変数を含むパターンと（変数を含まない静的な）ヘッドとの間のすべての解候補（各変数への）が洗い出されて「結果」として返されるため、上記「困難性Ａ」克服のための基礎的な機構をシンプルに与えることができ、推論過程の可視化や検証だけでなくさらにはルールの修正等といったユーザーの手を介した学習をも容易にする。
また、より単純かつ再帰的な（または相互再帰的な）同一構造を持つ部分問題へとシンプルに変換し、推論過程の表示や検証（完全性や健全性の）を容易にする。
また、変数を自然言語に埋め込む形式で表現するルールのヘッドが変数を含まない場合の健全な機械的解釈と、シームレスでわかりやすくかつシンプルで統一的な処理メカニズムと、これらに該当する推論過程の説明を与えることができる。

【００２０】
また、請求項１２は、
前記自動単一化処理又は自動導出処理において、
変数を含むパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
パターンとヘッド部の頭、尾又は両方に存在する文字数分の固定文字が一致していない場合に偽を返す、
請求項１から１１のいずれか一項に記載の処理方法を提供する。
これにより、一律の探索に比して、明らかに大幅な計算量節約と処理速度向上が図れる。

【００２１】
　また、請求項１３は、
　前記自動単一化処理又は自動導出処理において、
　変数を含む第一のパターンと変数を含むヘッド部をもつホーン節との単一化を行う場合、
　パターンとヘッド部の頭及び尾に存在する文字数分の固定文字が一致している場合、
　請求項５又は６に記載の処理方法における「サブゴール」として当該ホーン節のボディ部を適用した結果が真となる各束縛情報により当該ホーン節のヘッド部を束縛して得られる各第二のパターンについて、
　第二のパターンが変数を含まない場合に、
　　請求項２から４のいずれか一項に記載の処理方法における「パターン」として第一のパターンを用いかつ同処理方法における「定数文字列」として第二のパターンを用いて処理した結果を返す、
　請求項１から１２のいずれか一項に記載の処理方法を提供する。
　ここで、「第二のパターンが変数を含まない場合」とは、ホーン節（ルール）において、ヘッド部に含まれるすべての変数がボディ部「にも」含まれており、ボディ部のすべての変数が定数（固定文字列）で束縛された結果としてヘッド部のすべての変数も定数（固定文字列）で束縛（代入）されたことによりヘッド部の変数が消えてヘッド部が定数文字列と見なせるようになったことをいう。
　これにより、変数を含むパターンと変数を含むヘッド部をもつホーン節（ルール）との単一化をシンプルな計算モデルにより、ユーザーがトレースしやすい流れで機械的・自動的に行うことができる。
　すなわち、変数を含むパターンと変数を（各束縛情報により）含まなくなった（＝定数文字列化した）ヘッド部との間のすべての解候補（パターンの各変数へ代入すべきもの）を洗い出して「結果」として返す形に、元々のより複雑な問題（変数対変数の単一化）を帰着（問題を変換）させることにより、上記「困難性Ａ」克服のための基礎的な機構をシンプルに与え、思考プロセスとしても表現しやすいものにできている。また、この「帰着（問題の変換）」に当たって、より単純かつ再帰的な（または相互再帰的な）同一構造を持つ部分問題へとシンプルに変換しているため、推論過程の可視化や検証（完全性や健全性の）だけでなくルールの修正等といったユーザーの手を介した学習をも容易にする。
　結局、変数を自然言語に埋め込む形式で表現するルールのヘッドが変数を含む場合の健全な機械的解釈と、シームレスでわかりやすくかつシンプルで統一的な処理メカニズムと、これらに該当する推論過程の説明を容易に与えることができる。
【００２２】
　また、請求項１４は、
　変数を含む事実が入力されることをコンピュータが制限することを特徴とする
請求項１から１３のいずれか一項に記載の処理方法を提供する。
　ここで、変数を含む事実（ルールでない知識源）とは、例えば、「＄Ｘは＄Ｘである。」（トートロジー的に恒真となる事実）や「＄Ｘは＄Ｙである。」（表層的な文型だけが合致していれば真となってしまう根拠のない事実）等がある。
　これらがコンピュータに入力されることを制限することにより、パターン（変数を含みうる文）と事実（変数を含まない文（＝固定文字列））との自動単一化処理の計算が単純になり、自動処理による高速な回答が可能となる。また、根拠のない事実を知識源に混入しにくくする。
【００２３】
　また、請求項１５は、
　前記ルールがいわゆるホーン節と同様の形式であり、ヘッド部がボディ部に出現しない変数を含むことをコンピュータが制限することを特徴とする
請求項１から１４のいずれか一項に記載の処理方法を提供する。
　ここで、「ヘッド部がボディ部に出現しない変数を含むことをコンピュータが制限する

」とは、ヘッド部が「＄Ｘは素敵」であり、ボディ部が「＄Ｙは強い，＄Ｙは＄Ｚに優しい，＄Ｚは子供，＄Ｙは賢い」といったように、ボディ部のすべての文中の変数（この例では、＄Ｙと＄Ｚ）が束縛（代入）されても、ヘッド部の変数（この例では、＄Ｘ）が束縛されないまま残るようなルールが知識源として入力されたり、記憶されたり、起動されたりすることをコンピュータが制限することをいう。このようなルールは、事実の場合と同様に、ヘッド部に束縛されない変数を残すことにつながるため、健全でない推論結果を生み出しやすく、推論のスピードをも低下させる原因となる。

こうしたルールがコンピュータに入力されることを制限することにより、パターン（変数を含みうる文）とヘッド部（請求項１３により変数を含まない文（＝固定文字列）の集合に帰着可能）との自動単一化処理の計算が単純になり、自動処理による高速な回答が可能となる。また、根拠のない推論結果を生み出しにくくする。

【００２４】

また、請求項１６は、

前記自動単一化処理又は自動導出処理において、

束縛されない変数が互いに残っているパターンとヘッド部又は事実との自動単一化処理を、

固定文字列として極小となる解の集合を求めて返すことにより行うことを特徴する、

請求項１から１３のいずれか一項に記載の処理方法を提供する。

ここで、「固定文字列として極小となる解の集合」の例としては、「＄Ａは＄Ｂ」と「＄Ｃも＄Ｄ」との間の自動単一化の結果として、｛（＄Ａ＝”も”、＄Ｂ＝＜空＞、＄Ｃ＝＜空＞、＄Ｄ＝”は”），（＄Ａ＝空、＄Ｂ＝”も”、＄Ｃ＝”は”、＄Ｄ＝空）｝という２つの解を提供することがある。

これにより、無限個の解が存在する場合にも解の例（特に最小のシンプルな例）を提供することが可能になる。

【００２５】

また、請求項１７は、

前記自動単一化処理又は自動導出処理において、

束縛されない変数が互いに残っているパターンとヘッド部又は事実との自動単一化処理を、

変数を含む文字列として極小となる解の集合を求めて返すことにより行うことを特徴する、

請求項１から１３のいずれか一項に記載の処理方法を提供する。

ここで、「固定文字列として極小となる解の集合」の例としては、「＄Ａは＄Ｂ」と「＄Ｃも＄Ｄ」との間の自動単一化の結果として、｛（＄Ａ＝”＄Ｃも”、＄Ｂ＝＜自由＞、＄Ｃ＝＜自由＞、＄Ｄ＝”は＄Ｂ”），（＄Ａ＝＜自由＞、＄Ｂ＝”も＄Ｄ”、＄Ｃ＝”＄Ａは”、＄Ｄ＝＜自由＞）｝という２つの解を提供すること等がある。変数を束縛する値（変数に代入される値）の文字列長（ただし、変数が含まれる場合は変数の出現あたり１文字と見なす）の合計（解組みを通じた）が極小となるものと考えることができる。

これにより、無限個の解が存在する場合にも解の一般形を提供することが可能になる。

【００２６】

また、請求項１８は、

束縛されない変数が互いに残っているパターンとヘッド部又は事実との各文字列の頭及び尾に共通して存在する文字数分の固定文字が一致していない場合、

偽を返すことを特徴とする、

請求項１６又は１７に記載の処理方法を提供する。

これにより、単一化の可能性のない自動探索を節約し、自動処理を高速化することができる。

【００２７】

また、請求項１９は、

前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、

束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、

各変数に相手の文字列（第一の文字列中の変数には第二の文字列、第二の文字列中の変数には第一の文字列）のあらゆる部分文字列（＜空＞及び文字列全体も含む）又は自分自身を代入してみて両文字列が一致する場合を探す全探索により極小となる解の集合を求めることを特徴する、

請求項１６又は１７に記載の処理方法を提供する。

「又は自分自身を・・・代入」とは、文字列中の「＄Ｘ」には「＄Ｘ」そのものを代入することをいい、たまたま第一の文字列にも第二の文字列にも＄Ｘ（互いのスコープは異なっている）が含まれるときは、第一の文字列の＄Ｘと第二の文字列の＄Ｘをあえて区別できるように（例えば、「＄Ｘ１」と「＄Ｘ２」、等として）代入することによっても明らかに健全な計算を行うことができる。

これにより、健全性と停止性（計算の有限性）を確保しながら解集合の一定の網羅性（問題（パターン対）の性質によっては完全性）を確保することができる。

【００２８】

また、請求項２０は、

束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列そのものではなく、

束縛されない変数が互いに残っているパターンとヘッド部又は事実との各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除いて残る二つの文字列に相当する両文字列について、

請求項１９に記載の処理をする処理方法を提供する。

これにより、解集合の一定の網羅性（場合によっては完全性）を保ちながら全探索の空間を小さくでき、処理が高速化される。

例えば、「石川の＄Ａは＄Ｂが好き。」と「石川の＄Ｃも＄Ｄアイスが好き。」との間の自動単一化の過程として、そのまま解を探すのではなく、頭尾の共通部分を削除して

”＄Ａは＄Ｂ”

と

”＄Ｃも＄Ｄアイス”

とした上で、全探索により「＄Ａ」と「＄Ｂ」それぞれに、”＄Ｃも＄Ｄアイス”の全ての部分文字列（これは、＄Ｃや＄Ｄといった変数（＄｛・・・｝形式の変数も同じ）は１文字とすると、６文字であるため、６×（６＋１）／２＋１（＜空＞分）＋１（自分自身を代入分）＝２３通りある）を代入しつつ、同様に、「＄Ｃ」と「＄Ｄ」それぞれに”＄Ａは＄Ｂ”の全ての部分文字列（これは、同様に３文字であるため、３×（３＋１）／２＋１＋１＝８通りある）を代入してみてできる全ての両文字列（２３×２３×８×８通り）について、両文字列が一致した場合（例えば、（

”もはアイス”，

”はもアイス”，

”＄Ａはもアイス”，

”＄Ｃもはアイス”，

”＄Ａはも＄Ｄアイス”

※ちなみに”＄Ｃもは＄Ｂアイス”は一致しない

）となった場合の代入組の集合｛

（＄Ａ＝”も”、＄Ｂ＝”アイス”、＄Ｃ＝＜空＞、＄Ｄ＝”は”），

（＄Ａ＝＜空＞，＄Ｂ＝”もアイス”，＄Ｃ＝”は”，＄Ｄ＝＜空＞），

（＄Ａ＝”＄Ａ”，＄Ｂ＝”もアイス”，＄Ｃ＝”＄Ａは”，＄Ｄ＝＜空＞），

（＄Ａ＝”＄Ｃも”，＄Ｂ＝”アイス”，＄Ｃ＝”＄Ｃ”，＄Ｄ＝”は”），

（＄Ａ＝”＄Ａ”，＄Ｂ＝”も＄Ｄアイス”，＄Ｃ＝”＄Ａは”，＄Ｄ＝”＄Ｄ”）
｝又は、各代入組の部分的代入の集合（例えば｛
（＄Ａ＝”も”、＄Ｂ＝”アイス”），
（＄Ａ＝＜空＞，＄Ｂ＝”もアイス”），
（＄Ａ＝”＄Ａ”，＄Ｂ＝”もアイス”），
（＄Ａ＝”＄Ｃも”，＄Ｂ＝”アイス”），
（＄Ａ＝”＄Ａ”，＄Ｂ＝”も＄Ｄアイス”）
｝）
を解として答えることができる。
なお、＄Ａ＝”＄Ａ”や＄Ｂ＝”も＄Ｄアイス”は、＄Ａや＄Ｄが自由であり、そこにどのような文字列を代入しても単一化できる（すなわち、無限の解があるということと、その無限個の解（文字列）の制約（正規表現的な形）のあり方）を示すことができる。

【００２９】

また、請求項２１は、
各変数に相手の文字列のあらゆる部分文字列（＜空＞及び文字列全体も含む）又は自分自身を代入してみて両文字列が一致する場合を探す全探索により極小となる解の集合を求める際、
変数に代入する部分文字列に変数が含まれる場合に、当該部分文字列に含まれる変数を両文字列にそれまで存在しなかった新しい変数で置き換えながら、該新しい変数に対しての同様の代入も以後再帰的に行いながら両文字列が一致する場合を探す全探索を行うことを特徴とする
請求項１９又は２０に記載の処理方法を提供する。
これにより、自文字列内の固定文字を相手文字列内の変数を介して反射的に自文字列内の変数に代入した解をも探す必要のある性格の問題（例えば、両文字列＝｛”＄Ｘは＄Ｘ”，”い＄Ｙう”｝の場合では、解候補として（＄Ｘ＝”いう”，＄Ｙ＝”うはい”）を列挙できなければならない）について、解探索の健全性を維持したまま、解候補列挙の網羅性をより高めることができる。上の例では、両文字列（「パターン対」とも呼ぶ）＝｛”＄Ｘは＄Ｘ”，”い＄Ｙう”｝から（＄Ｘ＝”いう”，＄Ｙ＝”うはい”）という解候補を列挙できることを、本発明により再帰的に（＄Ｘ＝”い＄｛Ｙ１｝う”）⇒（＄｛Ｙ１｝＝＜空＞）⇒（＄Ｙ＝”うはい”）という探索枝（探索木の一部であり、本発明に沿った幅優先探索の場合は有限回で生成可能である）を生成しうることによって、示すことができる。
「再帰的に・・・全探索」とは、例えば、各変数への各ありうる代入（前の請求項と同じ）を再帰的に繰り返す幅優先探索をしながら一致する代入系列（例えば、上記の（＄Ｘ＝”い＄｛Ｙ１｝う”）⇒（＄｛Ｙ１｝＝＜空＞）⇒（＄Ｙ＝”うはい”）、等）を列挙することである。最初の代入のバリエーションは、両文字列中の変数の種類がそれぞれ２個と３個であり文字列の長さがそれぞれ６字と７字（変数はまとめて１字と数える）であれば、２個×（７×（７＋１）／２＋１＋１）＋３個×（６×（６＋１）／２＋１＋１）＝６０＋６９＝１２９通りとなる。これらの枝から派生するさらなる枝の最長深さが均等になるように幅優先探索を行う場合、代入回数等によるリミッター（上限値と比較するプログラムステップ等）を設けることにより、一回一回の探索を適当な計算量に制限することができる（再帰性によって保証されなくなった停止性や応答速度の確保）。なお、深さ優先探索とした場合にも、深さの制限や代入回数等によって計算量を制限することができる。いずれも自己再帰や相互再帰による無限ループを、再帰的な関数の呼び出しを管理するスタック構造（構造体の配列）等を設けて、先祖（より根方向）となる枝の呼出しパラメータ（その代入系列（代入文脈）に応じた両文字列の状態表現を含んでいてもよい）を現在の呼出しパラメータと比較したりすることにより回避することができる。また、両文字列の頭尾に存在する固定文字列が共通して存在する文字数分一致しているかどうかなどにより、単一化しえない枝の枝刈りを行うこともできる。なお、制限した計算量（例えば代入百万回や探索深さ２０以内）の探索により解が見つからなかった場合に「解なし（精

度：代入百万回かつ探索深さ２０以内）」として、見つかった場合に、「（＄Ｘ＝”いう”，＄Ｙ＝”うはい”），（・・・），・・・（但し、精度：代入百万回かつ探索深さ２０以内）」等として探索精度を明示しながらユーザに答えることもできる。また、代入の制限回数以内かつ制限探索深さ以内で明らかに全探索できた場合（完全性のある場合）には、その旨を「解なし（全探索済）」あるいは「（＄Ｘ＝”いう”，＄Ｙ＝”うはい”），（・・・），・・・（全探索済）」等と明示して、ユーザに答えることもできる。

【００３０】

また、請求項２２は、

変数に代入した結果についても、各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除いて残る二つの文字列を、

「相手の文字列のあらゆる部分文字列（＜空＞及び文字列全体も含む）」の対象となるの「相手の文字列」として、探索が進むごとに、文字列の頭又は尾に存在する固定文字数が少なくすることを特徴とする、

請求項１９～２１のいずれか一項に記載の処理方法を提供する。

これにより、探索を進めるごとに（先の枝にいくほど）枝の個数が少なくなり計算量を節約することができる。

【００３１】

また、請求項２３は、

前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、

束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、

各変数に、固定文字列（空文字を含む）又は変数を含む文字列を再帰的に代入してみて解の集合を求める処理方法であって、

代入してみる変数を相手文字列の頭又は尾が固定文字である自文字列の対応する頭又は尾の変数に限定し、

代入値を、該頭である場合には該頭の固定文字もしくは固定文字列に続く新しい変数又は＜空＞とし、該尾である場合には新しい変数に続く該尾の固定文字もしくは固定文字列又は＜空＞とし、

変数に代入した結果について、各文字列の頭及び尾に共通して存在する文字数分の固定文字を両者から除くことを繰り返す、

ことを特徴とする請求項１６又は１７に記載の処理方法を提供する。

これにより、頭及び尾の固定文字の存在に着目して解の列挙範囲を（探索の網羅性を損なわずに）大幅に制限すること可能になり、大幅な計算量節約と高速化が可能になる。

【００３２】

また、請求項２４は、

前記固定文字列又は変数を含む文字列として極小となる解の集合を求める際、

束縛されない変数が互いに残っているパターンとヘッド部又は事実とからなる両文字列（第一の文字列と第二の文字列とする）に含まれる各変数が、各文字列内では共通に束縛され、文字列間では独立に束縛されるものとしながら（あるいは、同名変数を共通に束縛するスコープを各文字列内のみ（あるいは、文字列間は変数のスコープ外）としながら）、

各変数に、固定文字列（空文字を含む）を再帰的に代入してみて解の集合を求める処理方法であって、

各文字列の頭及び尾に存在する文字数分の共通する固定文字を除いてできる両文字列中の固定文字種（変数の識別子やデリミタを除いた文字種）だけを用いてできる束縛情報を、束縛する値の文字列長の合計が小さいものから順に列挙していくことにより求める、

請求項１６又は１７に記載の処理方法を提供する。

これにより、一定の性質をもった問題については、効率よく解に到達し、列挙することもできる。

【００３３】

また、請求項２５は、

求めた「変数を含む文字列として極小となる解の集合」をもとに、

当該ホーン節の変数を当該変数を含む文字列で一時的に書き換えてできるホーン節により探索を継続する請求項１９～２４に記載の処理方法を提供する。

これにより、パターンとホーン節（ルール）のヘッド部とが単一化するために必要であることが計算された制約（上記の例では、変数「＄Ｃ」が”＄Ａは”という形の文字列で束縛（代入）されることでしか単一化しえないという制約の働くような単一化方針（例えば、上記最後の代入組の場合）もある）を、そのホーン節のボディ部の探索プロセスに伝播することができ、探索を効率化することができる。

【００３４】

また、請求項２６は、

求めた「変数を含む文字列として極小となる解の集合」をもとに、

「変数を含む文字列」の変数部分に、当該「変数を含む事実」の解組の集合としてあらかじめ定義された固定文字列組の集合の各要素を代入してできる固定文字列組の集合を返す、

請求項１９～２５のいずれか一項に記載の処理方法を提供する。

これにより、変数（一個又は複数）を含む事実とその変数に埋めうる適切な答え（又は仮説）の集合とをひもづけたデータベース（例えば、「言葉方程式（登録商標）」「ＶＬＡＮＫ（登録商標）」中の知識源）と効率的に連携した解探索を行うことができる。また、似た文型をもつ大量の事実を、本発明の枠組みと連携させて効率的に管理することができる。

【００３５】

また、請求項２７は、

前記自動単一化処理又は自動導出処理において、

変数を含む第一のパターンと変数を含む第二のパターンとの自動単一化を行う場合、

両パターンの頭及び尾に存在する文字数分の固定文字が一致しており、かつ、同じ変数が各パターン内で一度しか出現しない場合、

無限個の解の存在を意味する情報を返す、

請求項１から２６のいずれか一項に記載の処理方法を提供する。

【００３６】

また、請求項２８は、

請求項１から２７のいずれか一項の方法を実行するための装置を提供する。

【００３７】

また、請求項２９は、

請求項１から２７のいずれか一項の方法をコンピュータに実行させるためのコンピュータプログラムを提供する。

【発明の効果】

【００３８】

本発明の効果は、各請求項について上述したとおりである。

【図面の簡単な説明】

【００３９】

【図１】図１は本発明を最も効果的に実施するネットワークシステムの全体図である。（実施例１）

【図２】図２は本発明を実施するコンピュータのハードウエア構成を示した説明図である。（実施例１）

【図３】図３は本発明を実施する端末における画面表示の例を示した説明図である。（実施例１）

【発明を実施するための形態】
【００４０】
本発明は、ネットワークに接続された／接続されてないあらゆる種類のコンピュータ（クラウドサーバー、大型汎用機、デスクトップＰＣ、ノートＰＣ、携帯端末、本発明専用機）において実施することができるが、一般的なマルチタスク機能とウィンドウ機能をもちインターネットに接続されたＰＣにより実現した形態により説明する。
図１は本発明を最も効果的に実施するそのネットワークシステムの全体図であり、コンテンツ管理装置１は、利用者ＰＣ２又は利用者兼管理者ＰＣ２により、本発明で取り扱うコンテンツ（ルールや事実）の投稿を受け付け、所定のポリシーによって共有アクセス権やコンテンツ提供優先順位を制御しながらコンテンツを配信する公知のファイル共有サーバーでよい。利用者ＰＣ２は、その利用者ごとに、その信じる内容（や検討したい内容）を自然言語で表現したコンテンツ（ルールや事実）を本発明の方法に沿って入力して記憶し本発明の方法によって処理するコンピュータ、携帯端末等である。
図２は本発明で用いるコンピュータのハードウエア構成を示した説明図である。これは計算機の構成としては一般的なものであり、本発明の特徴は、不揮発性メモリ２ｈに記憶されたプログラム３及びコンテンツデータ４の指示や記載に従い自然法則を用いてＣＰＵ２ｅで処理される処理の内容、及び、利用者との対話プロトコル（書式）の在り方にある。
図３は本発明を実施する端末における画面表示の例を示した説明図である。図では「？＄Ａしない」という問い（オープンクエスチョン形式）に対して、図示しない記憶したリストＬの１行目の「されていやなことはなるべくしない」という事実を用いて「＄Ａ　＝　されて嫌なことをなるべく／１／」という答えを列挙し、同様にリストＬの９行目の「いじめをしない」という事実を用いて「＄Ａ　＝　いじめ／９／」と答えている。
【００４１】
このような実施形態においては、Ｃ＋＋言語等を介して製造されたコンピュータプログラムにより、コンピュータ（ＰＣ等）に可能なあらゆる動作を実施させることができる。したがって、自然法則を用いたコンピュータの公知の基本的な動作（レジスタやメモリやストレージへの記憶・取り出し、四則演算、比較、等）や既に開発環境等の標準ライブラリとなっている一般的な機能（変数やＮ次元配列への代入・比較・コピー、文字列や数値の操作／比較／表示、リスト構造・キュー構造・ハッシュ構造・ベクトル型等の操作、バイナリーサーチやソート操作、正規表現によるパターンマッチ操作、ネットワーク通信操作、等）の実現方法については、本発明の本質（新規性や進歩性につながる独自の構成）とも異なり、当業者にもよく知られているので、詳述はしない。一方、このような一般的な機能を組み合わせて、本発明に固有の処理をどのような操作順序や記憶の仕方（アルゴリズムやデータ構造）で実現するかについては、以下に詳述する。このアルゴリズム等の説明、上記した課題解決手段の説明及び本発明の図面の説明を提供して開発の参考とさせることにより、標準レベルの職業的プログラマであれば、標準的なアプリケーションプログラム開発技法（Ｃ＋＋によるＧＵＩアプリケーション統合開発環境、等）を用いて、本発明の方法をコンピュータに実施させることが明らかに可能となり、また、この方法のための装置やプログラムを作成することも可能となる。
【００４２】
以下に示すアルゴリズムは、Ｃ言語／Ｃ＋＋言語等を用いて所望の「関数」を新たに定義した上で関数呼出しを行うという標準的な構造化プログラミングをベースにしたアルゴリズムであり、当業者は、下記各関数の機能分担に応じた入出力や機能切り分けを行いながら（或いは、下記に示した仕様に沿って関数群をそのままコーディングすることにより）、本発明の解決手段に相当する処理手段（プログラムや計算機）を実現することが可能となる。
【００４３】

■問い合わせＰに対する知識リストＬの処理アルゴリズム仕様

==========
関数Ｑ１（Ｐ：変数を含みうるパターン，Ｌ：知識リスト）：★全請求項に共通するメイン関数の実施例１★
｛
１．Ｌの最初の行の内容Ｂ（事実又はホーン節）を取り出す。
ただし、Ｌが空の場合（最初の行が取り出せなかった場合）、偽を出力して終了。

２．変数の束縛情報Ｃの初期値を空とし、下記関数Ｈ（Ｐ，Ｂ，Ｃ）を呼出し、真となる（＝マッチする）束縛情報（＝解集合）Ｃを出力。
真（＝マッチした）の場合
真（はい）を出力する（マッチした内容Ｂの行番号も添えて出力する）。
偽の場合
何もしない。

３．Ｌの次の行の内容Ｂ’が
ある場合
・Ｂ←Ｂ’とし、２に移動する。
ない場合
・２で真を一度も出力していない場合は偽（いいえ）を出力。
・終了。
｝
【００４４】

■問い合わせＰに対する知識リストＬの処理アルゴリズム仕様（事前固定文字列化版）
==========
関数Ｑ２（Ｐ：変数を含みうるパターン，Ｌ：知識リスト）：★全請求項に共通するメイン関数の実施例２（事前固定文字列化版）★
｛
（事前の固定文字列化）
１．｛Ｌのみから演繹できるすべての文字列／その各文字列を演繹する根拠として用いたＬの行番集合｝を各行とする固定文字列テキストＴを求める。
具体的には、上記関数Ｑ１（”＄Ａ”，Ｌ）の出力系列（Ａ＝解１／根拠１，Ａ＝解２／根拠２，・・・）の要素を各行としてマージしたテキストＴを求める。
求める途中で無限ループになる場合について、明示的な呼出管理スタック等を用いた無限ループ検出機能により演繹を部分的に省略する。
・例えば、ホーン節の真偽の確認をある変数束縛条件下でコンピュータが行う途中で
同じホーン節の真偽を同じ変数束縛条件下で確認しようとしていないかを、
構造体（確認中のホーン節識別子，現在の変数束縛条件）をメンバーとする呼出管理スタックを管理し、
呼出し時にスタックの根に向かって同じ組み合わせがないかリニアサーチすることにより
無限ループを検出する。

（固定文字列のみである故にＱ１よりシンプルな方法（関数Ｆ）で解集合を求める）
２．テキストＴの各行ｔにつき、下記関数Ｆ（Ｐ，ｔ，Ｃ）が真となるＣをすべて出力する。

（最終結果の出力）

３．Ｃが一つでもあれば真を出力し、なければ偽を出力する。
}
【００４５】

■問い合わせＰに対する、一の事実（固定文字列）又はホーン節Ｂ（但し、ヘッドに含まれるすべての変数がボディにも存在）の処理アルゴリズム仕様
==========
関数Ｈ（Ｐ：変数を含みうるパターン，Ｂ：一の事実（固定文字列）又はホーン節，Ｃ：変数の束縛情報）：★請求項１～１５の発明で使用する関数の実施例１－１★

{
Ｐが変数を含まない場合｛
Ｂが事実である場合｛
Ｐ＝Ｂの場合は、真を返す。
そうでない場合は偽を返す。
}
Ｂがホーン節である場合｛
Ｂのヘッド部が変数を含まない場合｛
Ｐ＝Ｂのヘッド部の場合は、関数Ｄ（Ｂのボディ部，Ｅ）の真偽を返す。
そうでない場合は偽を返す。
}
Ｂのヘッド部が変数を含む場合｛
Ｆ（Ｂのヘッド部，Ｐ，Ｃ１）を真とする各Ｃ１について、
関数Ｄ（Ｂのボディ部にＣ１を適用したもの，Ｅ）の結果が真となるＣ１があれば、
真を返す。
そうでない場合は偽を返す。
}
}
}
Ｐが変数を含む場合｛
Ｂが事実である場合｛
Ｆ（Ｐ，Ｂ，Ｃ１）が真となるＣ１が存在すれば、そのようなすべてのＣ１をＣに加えて真を返す。
そのようなＣ１が存在しなければ、偽を返す。
}
Ｂがホーン節である場合｛
Ｂのヘッド部が変数を含まない場合｛
Ｆ（Ｐ，Ｂのヘッド部，Ｃ１）を真とするＣ１が存在する場合で、かつ
関数Ｄ（Ｂのボディ部，Ｅ）が真であれば、そのようなすべてのＣ１をＣに加えて真を返す。
そうでない場合は偽を返す。
}
Ｂのヘッド部が変数を含む場合｛

★請求項１２の発明を実施する場合★
／／高速化の工夫（オプション）開始
ＰとＢのヘッド部の頭及び尾に存在する文字数分の固定文字が一致していない場合｛
偽を返す。
｝／／高速化の工夫（オプション）終了

★請求項１３の発明の実施★
ＰとＢのヘッド部の頭及び尾に存在する文字数分の固定文字が一致している場合｛
関数Ｄ（Ｂのボディ部，Ｅ）の結果が真となるＥ中のいずれかの束縛情報Ｇについて、
関数Ｆ（Ｐ，Ｂのヘッド部をＧで束縛したもの（固定文字列になるはず），Ｃ１）を真とするＣ１があれば
そのようなすべてのＣ１をＣに加えて真を返す。
そのようなＣ１が存在しない場合は偽を返す。
}
}
}
}
}
【００４６】

■ホーン節のボディ部（又は変数を共有する重文の問い合わせ）の解集合の列挙アルゴリズム仕様
==========
呼出例
→関数Ｄ（"$Aと$Bは友達だ,$Aは人間,$Bは人間"，Ｅ);
戻り例
←真（Ｅ＝（（A=花子，B=太郎），（A=太郎，B=次郎），（A=次郎，B=三郎））
==========
関数Ｄ（Ｊ：ボディ部（又は変数を共有する重文の問い合わせ），Ｅ：解集合）：★請求項６の発明で使用する関数の実施例１－２Ａ★
※ただし、知識源全体Ｌはグローバル変数に格納して参照可能とする
｛
１．Ｊ中の最初の文Ｍを取り出す。最初の文が存在しない場合真を返す。

２．知識源全体Ｌに対して、前記関数Ｑ１（Ｍ，Ｌ）又はＱ２（Ｍ，Ｌ）を呼び出す。
結果が偽のときは
偽を返す。
真のとき
Ｍが変数を含まない場合
何もしない。
Ｍが変数を含む場合
Ｍを真とするためにとりうる束縛情報の集合Ｃを記憶する。
※真とするために束縛不要の変数は、その旨を明示的に示すこともできる。
※束縛値が無限集合でありながら真とするために一定の条件を要求する場合は、その条件を付記することもできる。

３．Ｊ中の次の文Ｍ’を取り出す。存在しない場合、真（Ｅ＝束縛情報の集合Ｃ）を返す。

４．知識源全体Ｌに対して、前記関数Ｑ１（Ｍ’，Ｌ）又はＱ２（Ｍ’，Ｌ）を呼び出す。
結果が偽のときは

偽を返す。
真のとき
Ｍ’が変数を含まない場合
何もしない。
Ｍ’が変数を含む場合
Ｍ’を真とするためにとりうる束縛情報の集合Ｃ’と記憶してあった集合Ｃとの積集合を新たなＣとする。
但し、元々の集合Ｃが空集合でなく積集合をとって空集合となったときは偽を返す。
※真とするために束縛不要の変数は、その旨を明示的に示すこともできる。
※束縛値が無限集合でありながら真とするために一定の条件を要求する場合は、その条件を付記することもできる。

５．３に戻る。

}

関数Ｄ２（Ｊ：ボディ部（又は変数を共有する重文の問い合わせ），Ｅ：解集合）：★請求項６の発明で使用する関数の別の実施例１－２Ｂ（関数の再帰呼び出しを用い束縛情報を他の文に波及させることにより高速化を図ったもの）★
{
１．Ｊ中の最初の文Ｍを取り出す。Ｍが存在しない場合真を返す。

２．知識源全体Ｌに対して、Ｑ１（Ｍ，Ｌ）又はＱ２（Ｍ，Ｌ）を呼び出す。
結果が偽のときは
偽を返す。
真のとき
Ｍが変数を含まない場合
Ｊの次の文がなければ真を返し
あれば次の文を新たなＭとし２に戻る。
Ｍが変数を含む場合
Ｊの次の文がなければ、とりうる束縛情報をＥに格納して真を返し
あればとりうる束縛情報を集合Ｃとして記憶する。

３．Ｅを空にし、集合Ｃ中の各束縛情報Ｋについて
再帰呼び出しＤ２（ＫによりＪの次の文以降を束縛してできるボディ部Ｊ’，Ｅ’）を実行する
結果が真のときは
Ｅに、真となったそのＫかつＥ’を加える

４．各Ｋについて一度も真となっていないときは偽を返し、一度でも真となっているときは真（Ｅ）を返す
}

関数Ｄ３（Ｊ：ボディ部（又は変数を共有する重文の問い合わせ），Ｅ：解集合）：★請求項５の発明で使用する関数の実施例１－３★
{
１．Ｊ中の最初の文Ｍ１を取り出す。最初の文が存在しない場合真を返す。

２．知識源全体Ｌに対して、Ｑ１（Ｍ１，Ｌ）又はＱ２（Ｍ１，Ｌ）を呼び出す。

結果が偽のときは
偽を返す。
真のとき
Ｍ１が変数を含まない場合
何もしない。
Ｍ１が変数を含む場合
とりうる束縛情報の集合Ｃ１を記憶する。

３．同様に、偽とならないうちはＭ２，Ｍ３・・・についてＣ２，Ｃ３・・・を求めていく。
偽となった場合は偽を返す。

４．残ったＣ１，Ｃ２，Ｃ３・・・Ｃｎについて積集合Ｃを求めて、真（Ｃ）を返す。
}

【００４７】

■問い合わせＰに対する事実Ｓの多長一致解（複数も可）の列挙アルゴリズム仕様
==========
呼出例
→関数Ｆ（"$Aと$Bは友達だ","太郎と次郎と三郎は友達だ", "");

関数Ｆ（Ｐ：変数を含みうるパターン，Ｓ：定数文字列，Ｃ：変数の束縛情報）：★請求項２の発明（ただし、「（又は最後）」と「又は前」を省いた実現例）で使用する関数の実施例１－４Ａ★
{
（準備）
１．Ｐ中の変数の種類数Ｖと、出現する変数の名前Ｎ１，Ｎ２，・・・を出現順に求める

（Ｐがもともと変数を含まない場合への対応）
２．Ｖが０のときは、
⇒　Ｐ＝Ｓのとき、真を返して終了。
⇒　Ｐ≠Ｓのとき、偽を返して終了。

（出力）
３．Ｖが１のときは、Ｎ１がＰ中に何回出現しようとも、ＰとＳの単一化の解Ｒ１は１つしかないので、
⇒　マッチする場合、その唯一の解を「ＣかつＮ１＝Ｒ１」として出力して、真を返して終了。
⇒　マッチしない場合、偽を返して終了。
※単一化そのものは、正規表現とのマッチ（同じ変数が二度以上出現する場合は後置参照あり）として実施可能

４．Ｖが２以上のとき
変数Ｎ１についての最長一致解Ｒ１で、Ｐ中の全てのＮ１を埋めてできるＰ’について再帰呼び出しＦ（Ｐ’，Ｓ，ＣかつＮ１＝Ｒ１）を実行

⇒　Ｎ１について最長一致解とした解集合が出力される

５．Ｒ１’←Ｒ１の最後の１文字を除いた文字列とし
Ｐ中の全てのＮ１をＲ１’で埋めてできるＰ’’について
再帰呼び出しＦ（Ｐ’’，Ｓ，ＣかつＮ１＝Ｒ１’）を実行
⇒　Ｎ１について、最長一致解より１文字除いた解候補Ｒ１’とした解集合が出力される

６．Ｒ１’が
１文字以上の場合、（★請求項３では最短一致解の文字数より大きい場合）
⇒Ｒ１←Ｒ１’として５に戻る。
０文字の場合、（★請求項３では最短一致解の文字数である場合）
終了。
}

【００４８】

関数Ｆ（Ｐ：変数を含みうるパターン，Ｓ：定数文字列，Ｃ：変数の束縛情報）：★請求項４の発明で使用する関数の実施例１－４Ｂ★
//Ｐ中で変数が最初に出現した直後の固定文字列の最初の文字Ｃに着目して高速化の工夫その１をした関数Ｆ
{
（準備）
１．Ｐ中の変数の種類数Ｖと、出現する変数の名前Ｎ１，Ｎ２，・・・を出現順に求める

（Ｐがもともと変数を含まない場合への対応）
２．Ｖが０のときは、
⇒　Ｐ＝Ｓのとき、真を返して終了。
⇒　Ｐ≠Ｓのとき、偽を返して終了。

（出力）
３．Ｖが１のときは、Ｎ１がＰ中に何回出現しようとも、ＰとＳの単一化の解Ｒ１は１つしかないので、
⇒　マッチする場合、その唯一の解を「ＣかつＮ１＝Ｒ１」として出力して、ＴＲＵＥを返して終了。
⇒　マッチしない場合、偽を返して終了。

４．（高速化の工夫その２。この処理は５以降でカバーされるため省いてもよい。★この工夫２は、請求項にはしていない★）
Ｖが２以上のときで、
すべての変数（Ｎ１（＝Ｎ），Ｎ２，・・・）について最長一致モードでマッチしてみた解と
すべての変数（Ｎ１（＝Ｎ），Ｎ２，・・・）について最短一致モードでマッチしてみた解と
が同一（Ｎ１＝Ｒ１，Ｎ２＝Ｒ２，・・・）のとき
⇒　その唯一の解を「ＣかつＮ１＝Ｒ１かつＮ２＝Ｒ２かつ・・・」として出力して終了する
そもそもマッチしなかった場合、
⇒　偽を返して終了。
同一でないとき

⇒　５に続く

５．Ｖが２以上のとき
変数Ｎ１についての最長一致解Ｒ１で、Ｐ中の全てのＮ１を埋めてできるＰ’について
再帰呼び出しＦ（Ｐ’，Ｓ，ＣかつＮ１＝Ｒ１）を実行
⇒　Ｎ１について最長一致解とした解集合が出力される

６．Ｐ中で変数Ｎ１の後すぐに
変数Ｎ１（自分自身）が続くとき
⇒　７以降で処理する
別の変数Ｎ２が続くとき
⇒　７以降で処理する（ただし、７で１文字ＣはＮ２の解Ｒ２の最初の１文字とする）
固定文字が続くとき
⇒　７以降で処理する

７．Ｐ中で変数Ｎ１が最初に出現した直後の固定文字である１文字Ｃ（直後が別の変数Ｎ２であるときはＲ２の最初の１文字Ｃ）が
最長一致解Ｒ１に含まれる場合
⇒　１）Ｒ１’←Ｒ１の文字列の最後からそのＣにあたるまでの部分文字列を削除した文字列
⇒　２）Ｐ’’←Ｒ１’でＰ中のすべてのＮを埋めてできるパターン
⇒　３）再帰呼び出しＦ（Ｐ’’，Ｓ，ＣかつＮ１＝Ｒ１’）を実行
⇒Ｎ１について一段階だけ短めの解が存在すれば出力できるはず
最長一致解Ｒ１に含まれない場合
⇒　Ｎ１についてはより短い解の列挙が終わったので、終了。

８．Ｒ１←Ｒ１’として７に戻る。
}

==========

【産業上の利用可能性】
【００４９】
本発明は、中学校前後から大人までにおける述語論理学自体の教育、一般の知識・情報の整理・交換・共有、事実（主張も含む）とルールからなる集合（哲学やマインドセット）の妥当性の検証に用いることができる。
【符号の説明】
【００５０】
１　コンテンツ管理装置
２　利用者ＰＣ又は利用者兼管理者ＰＣ
３　本発明によるプログラム
４　コンテンツデータ（ルールや事実の集合）

【図１】

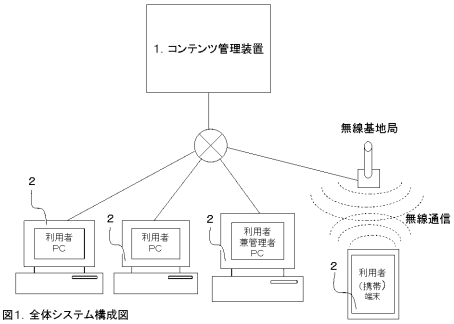

図1. 全体システム構成図

【図２】

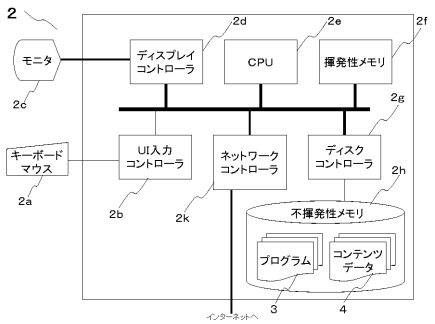

図2. ハードウエア構成図

【図３】